Generation of topologically useful entangled states

NEIL B. LOVETT*, BENJAMIN T. H. VARCOE†

School of Physics and Astronomy, University of Leeds, Leeds, LS2 9JT, United Kingdom

Measurement based quantum computation requires the generation of a cluster state (quantum resource) prior to starting a computation. Generation of this entangled state can be difficult with many schemes already proposed. We present an abstract scheme which can create 2D cluster states as a universal resource for quantum computing. We find a linear scaling of grid size with cluster depth. The scheme is also capable of creating more exotic topologies including 3D structures and the unit cell for topological error correction. We note its relevance to the cavity QED scheme in [30] although it could be applied to various architectures.

Key words: Quantum computation, cluster states, topological error correction

1 INTRODUCTION

The promise of quantum computation to provide computation fundamentally faster than current classical computers is dependent on finding both a scaleable architecture, which is experimentally feasible, and also useful algorithms. Much progress has been made over the years since Feynman, [14], originally proposed the idea of computing using quantum mechanical laws. Shor's algorithm for factoring numbers with an exponential speed up, [27],

* email: pynbl@leeds.ac.uk
† email: B.Varcoe@leeds.ac.uk

and Grover's for searching an unsorted database with an quadratic speed up, [16], are probably the two most important. Work on quantum algorithms has progressed rapidly especially with the introduction of algorithms based on quantum random walks [2, 4, 13], the most notable being the search algorithm of Shenvi, Kempe and Whaley [26] and an algorithm for element distinctness from Ambainis [3]. In fact, the quantum walk is now a standard technique in quantum algorithm design having recently been proven to be universal for computation [6, 20]. However, designing and building a physically realisable system has been slower due to the degree of control required to maintain coherent systems. Some important progress has been made in varying architectures including ion traps, solid state and optical systems, [9, 29, 12]. One of the hardest challenges experimentally is to create and maintain entanglement at varying points throughout the computation.

Cluster state quantum computation, [24], is a different paradigm in quantum computation than the circuit model. It is also known as measurement based quantum computation (MBQC). In MBQC, a general graph state is produced and then each site is entangled with its neighbour(s) by a controlled phase operation (C-Phase), eq. (1), entangling the two together as

$$C_{phase} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}. \tag{1}$$

In the case of two qubits they are firstly prepared in the $|+\rangle$ state,

$$\begin{aligned} |\psi\rangle &= \frac{1}{\sqrt{2}}(|0_1\rangle + |1_1\rangle) \otimes \frac{1}{\sqrt{2}}(|0_2\rangle + |1_2\rangle), \\ |\psi\rangle &= \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle). \end{aligned} \tag{2}$$

The C-Phase entangling operation is then applied between the two qubits to leave the resultant entangled state,

$$C_{phase}|\psi\rangle = \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle - |11\rangle). \tag{3}$$

Single qubit rotations and measurements are used to progress the computation. These measurements are based on the previous results which are fed forward. As all the entanglement is generated when the graph state is produced, no entanglement needs to be generated during the computation, which

experimentally is challenging. A cluster state is just a specific graph state, a square lattice. When first introduced it was hoped that this lack of ad-hoc entanglement would mean the experimental implementation would be much easier. However, this has not been the case, although work by Rudolf et al. has shown it to be feasible, [31]. Since its conception, many other schemes for cluster state preparation have also been introduced, [7, 21, 8, 17].

Recently, there has been a stimulus in work on cluster state generation using photons, [11, 28, 10, 18], and also topological error correction in cluster states, [22, 25, 23, 5, 15]. The photonic module, [11], is essentially a 'plug and play' cluster state generator. It can deterministically create cluster states and also uses the mobility of the photons to enable any output to be connected to any input. In addition to generating cluster states, another focus of the work is on topological error correction. Raussendorf and Goyal introduced the concept of using multiple qubits to encode one logical qubit in an attempt to make it fault tolerant, [23]. This is in contrast to the traditional cluster state in which the qubits are not protected from loss channels or errors in the system.

Due to the recent work on topological error correction and the idea of MBQC, the motivation for this work was to develop a scheme to produce a universal resource for MBQC. We also wanted a way to prepare these states which could be scaled up easily. We numerically modelled states we could theoretically prepare. We found we could create many interesting topologies with various possible applications. The recent work by Elham Keshafi et al. [19] on ancilla driven quantum computing has unusual 'twisted graph states' as a resource which our scheme could be used to create. The unit cell for topological error correction, [25], can also be created (with some Z measurements to remove qubits). As a cluster state (or graph state) is just a mathematical object, we review some basic graph theory used here in Sec. 2. We then introduce our basic scheme and the states we are able to prepare in Sec. 3. We develop it further in Sec. 4 before discussing its benefits and applications.

2 GRAPH THEORY

A general graph is an ordered pair, $G = (V, E)$, where V is a set of vertices and E is a set of edges. The edges in the set E are all described as unordered pairs relating two vertices, $(e_1, e_2) \in V$. The number of vertices in a graph, $|V|$, is the order of the graph and the number of edges, $|E|$, is it's size. In this, the most general description for a graph, the edges are unordered and

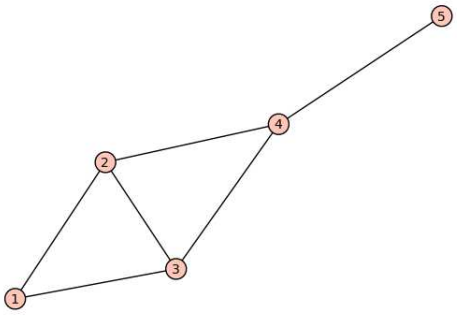

FIGURE 1
An example of a simple general graph with five vertices and six edges. It is undirected, unweighted, connected and has vertices of varying degree.

as such there is no orientation to the graph. This is known as an undirected graph, whereas one with an ordered pair of edges is a directed graph, where a specific direction is given by the edge. The graphs we produce in our scheme are all undirected such as the one shown in fig. 1. In the scheme we propose (detailed in Sec. 3), we assume the C-Phase operation, eq. (1), can be implemented perfectly and that each pair of qubits is maximally entangled. As such, the edges in our graphs will therefore all be the same, or of the same weight. In this case, our graphs are unweighted. However, if there was some error introduced and some pairs were less entangled than others, this could be modelled by turning our graphs into weighted ones. Weighted graphs have some weighting attached to each edge, which can then be interpreted in different ways. In some optimization problems, this could be a distance or time to travel from vertex to vertex. All the graphs we produce are known as simple graphs. A simple graphs is an undirected, unweighted graph with no self loops (an edge starting and ending at the same vertex) and a maximum of one edge between any two distinct vertices.

One important feature of graph theory we will use in this work is the notion of connectivity. A graph is said to be connected if there is a sequence of edges from any vertex to any other vertex. In a diagram, a disconnected graph would look like two separate entities. However, the set of vertices and edges of all parts are considered one graph. In our scheme, this is important as it determines whether we have a single copy of the graph produced or multiple

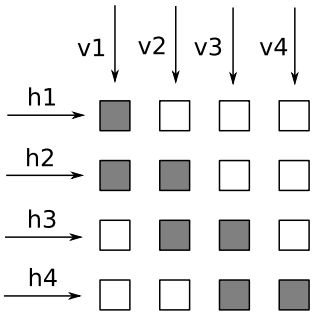

FIGURE 2
Structure we use to create our graph states. We imagine vertices moving in the direction of the arrows advancing by one site in the grid for each timestep. A dark site indicates it is active and so an edge is formed when two vertices pass in the same timestep.

ones. The degree of a vertex refers to the measure of adjacent connectivity. The number of edges incident on a vertex is the degree or valency of a vertex. This degree relates to the number of connections made between vertices in the graphs produced in our scheme. Cluster states are just a specific type of simple graph in which only nearest neighbours are connected. This creates a 2D lattice where each internal vertex is of degree four.

3 SCHEME

We now present our scheme for the generation of graph states which would be useful in quantum information processing. The scheme we describe is abstract and we do not initially define an architecture in which this could be physically realised. We show some of the useful states we can produce (by numerical simulation) and then discuss the drawbacks of this scheme.

3.1 Scheme

Consider a small grid of sites as in Figure 2. We imagine vertices of a graph moving across this grid horizontally and vertically by one site in the grid for each arbitrary timestep. If two of these vertices meet at a site we say a link or edge is formed between them. This edge represents the C-Phase entanglement generated between the two vertices which represent qubits. The vertices enter the grid in the direction of the arrows as a stream of atoms, one

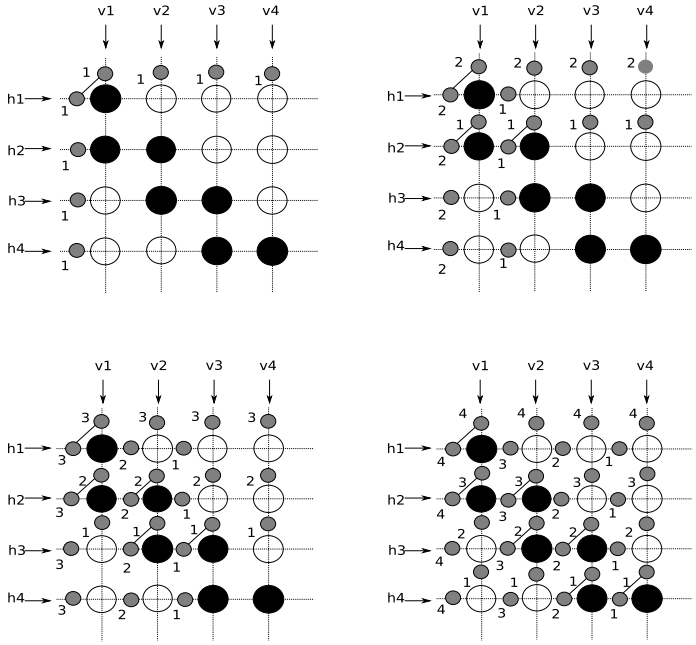

FIGURE 3
First four timesteps of a 4x4 grid. The vertices enter in the direction of the arrows. The black circles indicate active collision sites wheras white indicates the site is switched off.

entering for each arbitrary timestep. We extend this further and say that each of these sites could be active or inactive, forming an edge only when the site is active. An active site is considered to perform the C-Phase operation between the two vertices (qubits). Therefore instead of creating edges between static vertices, we build our graph states by moving the vertices through a set of 'edge joining' interaction regions, which can be switched on or off. This could be a grid of collisional cavities through which atomic beams are passed and entangled together as proposed by Blythe and Varcoe, [30].

As we have introduced the concept of the vertices moving with each timestep, we must also address the notion of when (in time) a vertex enters the grid. We do this by assigning a generation to each timestep. Using the grid in fig. 2 the vertices entering the first row of the grid will be labelled as $h_1g_4 - h_1g_3 - h_1g_2 - h_1g_1$. This implies that generation 1 passes into the grid first and so

therefore after four timesteps would be at the site in the top right hand corner of the grid. Using this labelling, it is easy to keep track of which generations of vertices are interacting together. We show how the vertices interact with the active sites and different generations in fig. 3. The mix of generations forming edges is obviously highly dependent on the size of the grid and also which sites are active. This dependency and our ability to change these factors allows various different structures to be created. Some of the structures produced by numerical simulation are shown next along with the grid pattern of active sites required to produce them.

Due to the vertices moving across the grid as a stream, we find that we get a number of the same structures created. The actual number produced depends on which sites are active and the number of timesteps taken. This scales as $t - (\sqrt{N} - 1)$ where t is the number of timesteps and N is the number of sites in the grid. However, some of these structures are incomplete. These come about from the vertices that have only partially traversed the grid when the number of timesteps are completed. These multiple and incomplete structures are shown in fig. 4. In the work by Blythe and Varcoe, [30], the sequence of atoms is pulsed at specific times to allow the creation of one structure as opposed to a continuous stream here.

3.2 States produced

We show several different structures we have produced by numerical simulation in order to show the variety of topologies our scheme can produce. All show the structure produced and the grid of active sites required to produce it. The number of timesteps was 10 for all of the examples. We can see the creation of a cluster state for universal quantum computing in figs. 5, 6 and 7. In fig. 8 we see additional links from a central structure. These could be used as ancilla qubits for algorithmic or error correction purposes. Finally in figs. 9 and 10 we see the initial unit cell of both cubic and and hexagonal lattices. If all sites are active we obtain the most highly connected state possible in this scheme. This is a cube with its corners connected diagonally as shown in fig. 11. This is in essence a superposition of all other structures that could be produced. Other more exotic structures can be created using a different combination of active sites.

3.3 Drawbacks

In the examples we have shown it is clear that each structure created can only have a maximum of eight vertices due to the size of the grid. Obviously the grid is not static in size, it can be be a square of any dimension N, $\sqrt{N} \times$

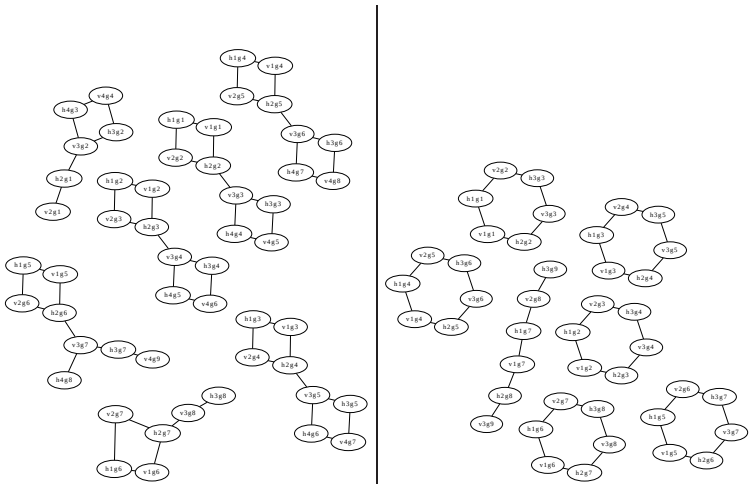

FIGURE 4
Multiple structures produced after 10 timesteps. The incomplete structures can easily be seen. Left - Full set of structures produced from the grid in fig. 6. Right - Full set of structures produced from the grid in fig. 10.

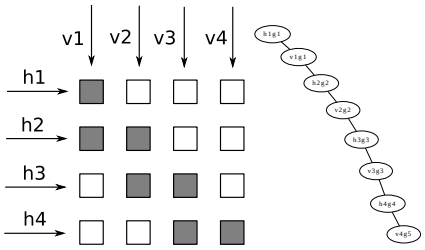

FIGURE 5
Grid of active sites and structure produced. The structure produced is a 1D lattice.

$\sqrt{N}$. We find that as the grid increases in size, the number of vertices in each structure also increases. The number of vertices present in any structure created can only be a maximum of $2\sqrt{N}$. Similarly, the maximum number of edges a vertex can form is $\sqrt{N}$, depending on how many active sites a vertex passes through. This restricts both the size and the topology of the structures we are able to create. However, as the grid increases in size we do not have

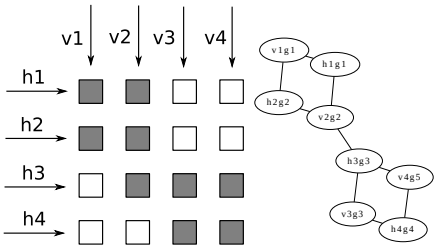

FIGURE 6
Grid of active sites and structure produced. The structure produced is the formation of a 2D lattice but not all the bonds have been formed.

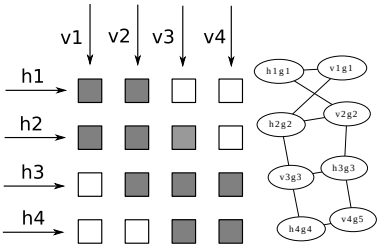

FIGURE 7
Grid of active sites and structure produced. The structure produced is a 2D lattice of depth two.

to stick to a specific pattern across the entire grid. Another option is to repeat an existing one on the diagonal. For example, if we have an 8 x 8 grid we could have any one of the patterns from figs. 5 to 10 repeated twice on the diagonal. This will create additional structures of the same form which could then be linked to form larger structures. This can be achieved by activating the adjacent skew diagonal elements between the repeated pattern. This is shown in figs. 12 and 13 and it is clear that any of the previous structures described could be linked in this way. The only thing that would limit the number of structures we could link together would be the size of the grid that could be constructed. However, as the grid is expanded in this way there are many inactive sites meaning it will probably not scale particulary well in a physically realisable device.

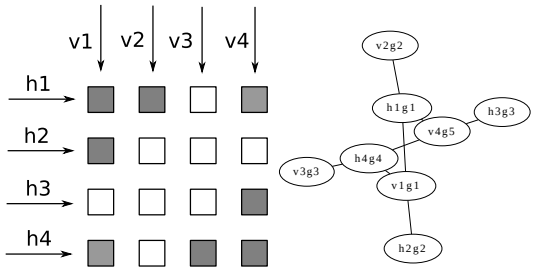

FIGURE 8
Grid of active sites and structure produced. The structure produced is a square lattice with an additional link on each vertex.

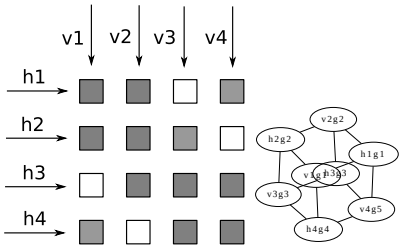

FIGURE 9
Grid of active sites and structure produced. The structure produced is a cube.

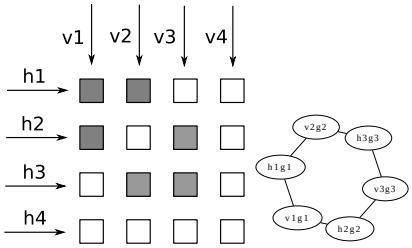

FIGURE 10
Grid of active sites and structure produced. The structure produced is a single cell of a hexagonal lattice.

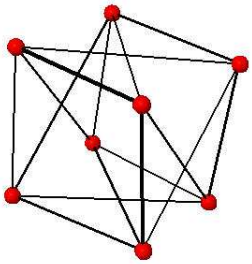

FIGURE 11
Structure produced when all sites are active. This is the most highly connected structure that can be produced from this model.

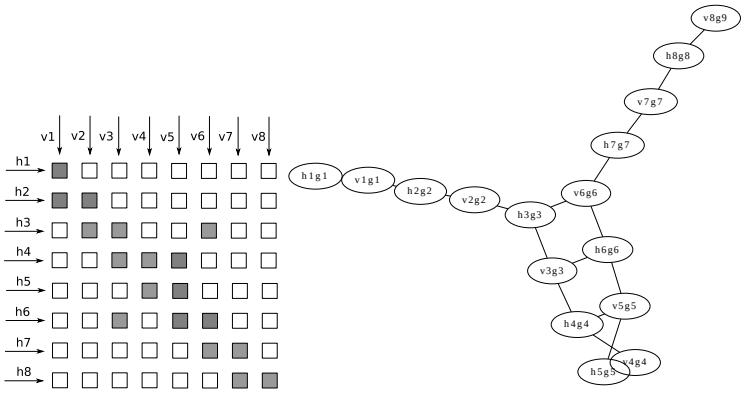

FIGURE 12
Grid of active sites and structure formed for the pattern in fig. 6 repeated and ‘linked’ together. If the number of timesteps was increased then more of the structure would link together at each timestep. The grid has two patterns which would create a 1D lattice, the skew diagonals link the two to form the two depth 2D lattice.

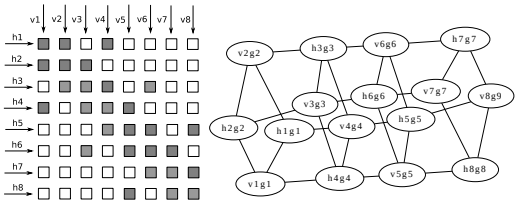

FIGURE 13
Grid of active sites and structure formed for the pattern in fig. 8 repeated and 'linked' together. The grid has two patterns which would create cubes, the skew diagonals link the two to form a 'chain' of cubes.

4 EXTENDED SCHEME

We now modify our scheme to allow the creation of larger structures using the same initial grid. We show some of the useful states we can produce (by numerical simulation).

4.1 Scheme

In order to solve some of the drawbacks in the initial scheme, we amend it slightly allowing the construction of much larger structures. In the extended scheme, we allow the vertices (qubits) to enter from either side of the grid in any pattern. This is shown in fig. 14 which shows the grid 'filling' up for the first few timesteps. Due to the change in where and when the vertices meet, we find the structures created are in effect 'infinite' in length, the only limiting factor is the number of timesteps the system is run for (assuming we can keep the system coherent for this time). This would allow any number of computational steps to be performed on the qubits. If we stick to creating just a 2D lattice (cluster state) then the size of the grid increases linearly with the depth of the cluster produced. This is clearly shown next where we show some of the examples of states produced, again by numerical simulation. The size of the grid compared to the structure produced is now much smaller than the original scheme.

Allowing the vertices to enter the grid in this fashion means that only four copies of the same structure are obtained. As is shown by the examples, the structures are much more similar in comparison to the original scheme. As only four structures are produced for any grid pattern it means that the connections are formed much faster. This ensures much of the structure is always complete and it is just the start and end which has less connections.

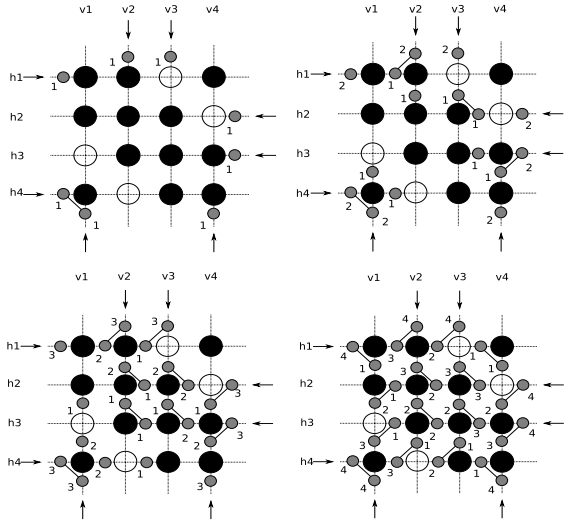

FIGURE 14
First four timesteps of a 4x4 grid as it 'fills up'. The vertices enter in the direction of the arrows. The black circles indicate active collision sites wheras white indicates the site is switched off.

This is shown in fig. 15 and is due to two factors. The first is due to the grid filling up at the start of the generation of the cluster which can be solved by starting the computation $\sqrt{N}$ steps later. The second is at the end of the structure which is incomplete as some of the vertices have not passed through the entire grid when we reach the number of required timesteps. This can be solved by ensuring the number of timesteps is an additional $\sqrt{N}$ than required for the computation. Therefore, overall we have a constant overhead of $2\sqrt{N}$ timesteps to add to any computation.

Our initial motivation was to generate a scaleable scheme for cluster state generation. We also found that we could easily create much more interesting structures as in the original scheme. If certain sites are activated we find the structure can loop and join itself again creating 3D structures and 'rings'. Other options are to repeat the basic pattern of the grid down the diagonal and then link them together. This could be used to form a 'ring of rings' for example. One interesting thing about copying the pattern in this sense is that we can create a 'ring' of any number of other structures. These individual structures can also have any number of vertices linked together.

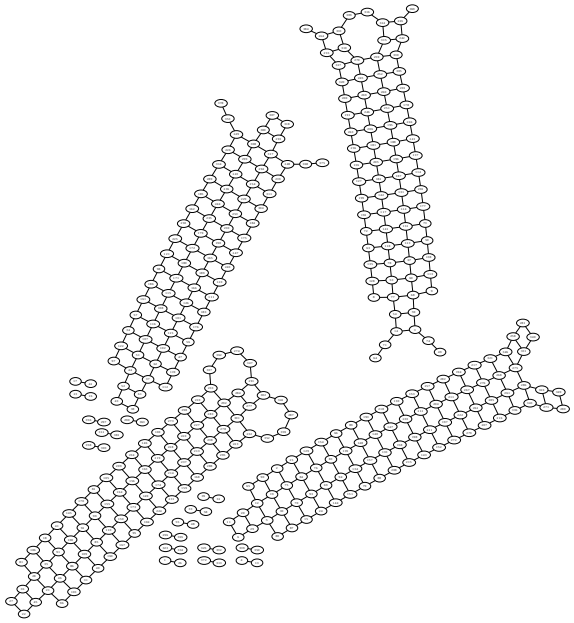

FIGURE 15
An example of the four structures created. The structures show the incomplete parts at the start and end.

4.2 States produced

We show several different structures we have produced by numerical simulation in order to show the variety of topologies our extended scheme can produce. All show the structure produced and the grid of active sites required to produce it.

We can see the creation of a cluster state for universal quantum computing in figs. 16, 17 and 18 and can clearly see the linear scaling of the grid with cluster depth. Figures 19 and 20 show how activating specific sites allows depth four or eight clusters to loop around to form a cube or octagon respectively. The 3D structures shown in figs. 19 and 20 can be repeated in a larger grid and then linked together in a similar way. This then achieves a ‘ring of rings’ topology where the number of vertices in either ring is just dependent on the size of the initial grid.

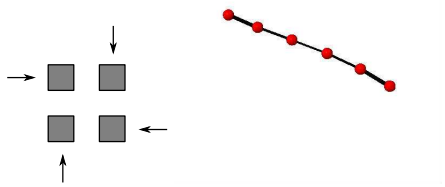

FIGURE 16
Grid of active sites and structure produced. 1D cluster with no restricted length.

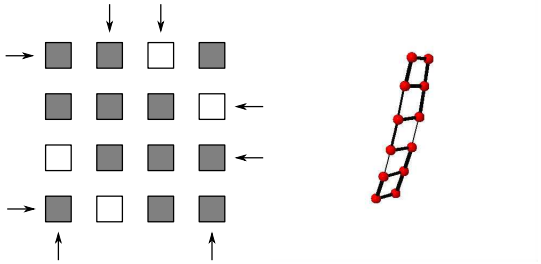

FIGURE 17
Grid of active sites and structure produced. 2D cluster of depth two with no restricted length.

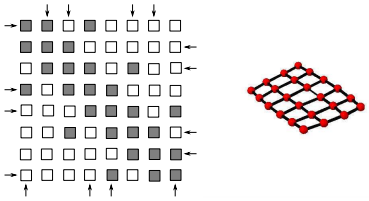

FIGURE 18
Grid of active sites and structure produced. 2D cluster of depth four with no restricted length.

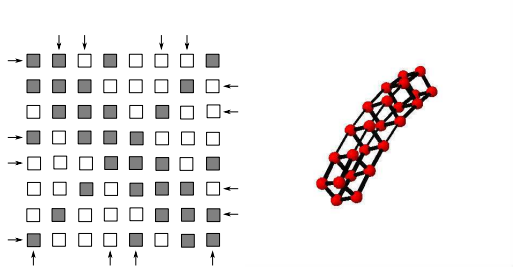

FIGURE 19
Grid of active sites and structure produced. 2D cluster of depth four looped round to form a cube of no restricted length.

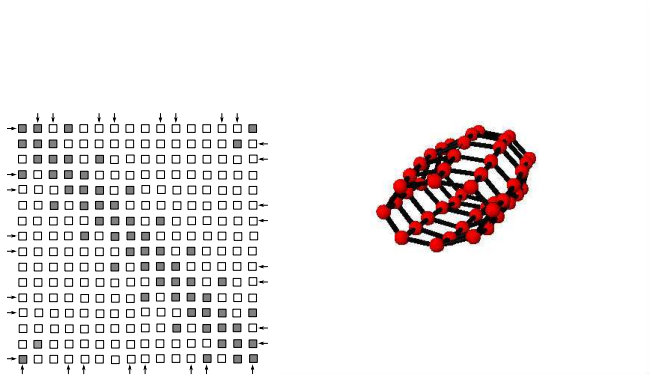

FIGURE 20
Grid of active sites and structure produced. 2D cluster of depth 8 looped round to form an octagon of no restricted length.

5 DISCUSSION / FURTHER WORK

We have presented a scheme to allow the creation of graph states. Our scheme could be applied to various architectures, however we do note its relevance to the cavity QED scheme previously mentioned, [30]. In this way we envisage the vertices as atoms and the edges of the graph as entanglement between these atoms. This architecture would seem to lend itself to quantum information processing applications. The basic cluster state produced by the extended

scheme is a universal resource for measurement based quantum computing. The scaling here is better than many other schemes that have been proposed. We only need to double the size of the grid to get a structure of double the depth (double the qubits in the cluster state). It is clear that doing this will create many inactive collision sites and as such is wasteful. However, if experimentally implemented these inactive sites need not be implemented, instead just creating the active zones and controlling the timing of the atoms is sufficient. Obviously, maintaining the timing and coherence of the qubits during the computation would still represent a significant challenge. As the computation can be run for an arbitrary time, the length of the structure is in effect 'infinite' as long as the system remains coherent. The unit cell for topological error correction can also be created (with some Z measurements to remove qubits).

In the scheme, we have assumed that when two vertices pass each other at an active site a full link or edge is always formed. This is an ideal case and we intend to amend our numerical simulations to include a probability of an edge forming. We would imagine there to be some critical probability similar to percolation theory where the structures are formed / not formed. This could mean the structures could be used in percolation problems when studying stuctures with broken or missing links. We notice that as there are multiple copies of the same structure it is unlikely that the same bonds will form in each one if there is chance of an error. As such we may be able to use some form of 'majority rules' or entanglement distillation scheme, [1], to ensure we have a complete structure. This would however remove many of the additional structures that are in effect created for 'free'.

The additional structures could easily be used to our advantage. This could be one of two options - multiple copies running either the same computation or different parts of a program. If we ran the same program on all the copies this would give a higher probability of success if there was a possibility of error as discussed above. It would also mean a probabilistic algorithm would have to be run less times as we would be in effect running it four times in one run. The other option would be to use each structure as a 'thread' in a multithreaded quantum computer. Activating specific sites in the grid at certain times would then allow the connection of the structures temporarily to allow communication between threads. This would be much more complex than the previous option but could also give benefits such as speed for example. The main problem with the multithreading idea would be communication between threads. This would be in the timing of the links between structures to pass information from one structure to another. It would also require longer

coherence times as the states may need to be 'stored' temporarily if communication is blocked with another structure to avoid a read-write error between threads in the same way as classical multithreading.

We intend to extend this work to provide a more physical setting in which we can discuss errors and implementation more thoroughly. The scheme we present here can not produce every structure as the vertices (qubits) cannot be directed from output to input as in [11]. We will address this in a specific architecture. Another possibility of the scheme is useful applications of the more exotic structures shown in the extended scheme. This would most likely be in the area of topological error correction mentioned briefly above. The unit cell can be created which is the basis of the error correcting schemes introduced by Raussendorf and Harrington [25]. An extension of error correction in this scheme would be to use multiple qubits in order to encode one logical qubit. This redundancy allows for fault tolerance but how many qubits we could use would obviously depend on how many qubits could physically be realisable.

Acknowledgments: NL is funded by the UK Engineering and Physical Sciences Research Council. BV is supported by the UK Engineering and Physical Sciences Research Council through an Advanced Fellowship GR/T02331/01 and Grant GR/S21892/01

REFERENCES

[1] J. I. Cirac A. Acin and M. Lewenstein. (2007). Entanglement percolation in quantum networks. *Nature Physics*, 3:256–259.

[2] D. Aharonov, A. Ambainis, J. Kempe, and U. Vazirani. (2001). Quantum walks on graphs. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 50–59. ACM.

[3] A. Ambainis. (2004). Quantum walk algorithm for element distinctness. In *Foundations of Computer Science, 2004. Proceedings. 45th Annual IEEE Symposium on*, pages 22–31. IEEE.

[4] A. Ambainis, E. Bach, A. Nayak, A. Vishwanath, and J. Watrous. (2001). One-dimensional quantum walks. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 37–49. ACM.

[5] Sergey Bravyi and Robert Raussendorf. (2007). A fault-tolerant one-way quantum computer. *Physical Review Letters*, 76:022304.

[6] A. M. Childs. (2009). Universal computation by quantum walk. *Physical Review Letters*, 102(18):180501.

[7] Jaeyoon Cho and Hai-Woong Lee. (2005). Generation of Atomic Cluster States through the Cavity Input-Output Process. *Phys. Rev. Lett.*, 95:160501.

[8] S. Fritzsche D. Gonta and T. Radtke. (2009). Generation of two-dimensional cluster states by using high-finesse bimodal cavities. *Phys. Rev. A.*, 79:062319.

[9] C. Monroe D. Kielpinski and D. J. Wineland. (2002). Architecture for a large-scale ion-trap quantum computer. *Nature*, 417:709.

[10] S.J. Devitt, A.G. Fowler, A.M. Stephens, A.D. Greentree, L.C.L. Hollenberg, W.J. Munro, and K. Nemoto. (2009). Architectural design for a topological cluster state quantum computer. *New Journal of Physics*, 11:083032.

[11] S.J. Devitt, A.D. Greentree, R. Ionicioiu, J.L. OBrien, W.J. Munro, and L.C.L. Hollenberg. (2007). Photonic module: An on-demand resource for photonic entanglement. *Physical Review A*, 76(5):52312.

[12] R. Laflamme E. Knill and G. J. Milburn. (201). A scheme for efficient quantum computation with linear optics. *Nature*, 409:46.

[13] E. Farhi and S. Gutmann. (1998). Quantum computation and decision trees. *Physical Review A*, 58(2):915–928.

[14] R. P. Feynman. (1986). Quantum mechanical computers. *Foundations of physics*, 16(6):507–531.

[15] Austin G. Fowler and Kovid Goyal. (2009). Topological cluster state quantum computing. *Quant. Info. Comput.*, 9:721–738.

[16] L. K. Grover. (1996). A fast quantum mechanical algorithm for database search. *Proceedings of the twenty-eight annual ACM symposium on Theory of computing*, page 212.

[17] Kilian Singer Harald Wunderlich, Christof Wunderlich and Ferdinand Schmidt-Kaler. (2009). Two-dimensional cluster-state preparation with linear ion traps. *Phys. Rev. A.*, 79:052314.

[18] Radu Ionicioiu and William J. Munro. (2009). Constructing 2D and 3D cluster states with photonic modules. arXiv:0906.1727v2 [quant-ph].

[19] E. Kashefi, DKL Oi, D. Browne, J. Anders, and E. Andersson. (2009). Twisted Graph States for Ancilla-driven Universal Quantum Computation. *Electronic Notes in Theoretical Computer Science*, 249:307–331.

[20] N. B. Lovett, S. Cooper, M. Everitt, M. Trevers, and V. Kendon. (2010). Universal quantum computation using the discrete-time quantum walk. *Physical Review A*, 81(4):42330.

[21] Ming Yang Ping Dong, Zheng-Yuan Xue and Zhuo-Liang Cao. (2006). Generation of cluster states. *Phys. Rev. A.*, 73:033818.

[22] J. Harrington R. Raussendorf and K. Goyal. (2006). On measurement-based quantum computation with the toric code states. *Annals. of. Physics.*, 321:2242.

[23] J. Harrington R. Raussendorf and K. Goyal. (2007). Topological fault-tolerance in cluster state quantum computation. *New. J. Phys.*, 9:199.

[24] R. Raussendorf and H. J. Briegel. (2001). A One-Way Quantum Computer. *Phys. Rev. Lett.*, 86:5188–5191.

[25] R. Raussendorf and J. Harrington. (2007). Fault-Tolerant Quantum Computation with High Threshold in Two Dimension. *Phys. Rev. Lett.*, 98:190504.

[26] N. Shenvi, J. Kempe, and K. B. Whaley. (2003). A Quantum random-walk search algorithm. *Physical Review A*, 67(5):52307.

[27] P. W. Shor. (1997). Polynomial-time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM. J. Comput.*, 26:1484.

[28] A.M. Stephens, Z.W.E. Evans, S.J. Devitt, A.D. Greentree, A.G. Fowler, W.J. Munro, J.L. OBrien, K. Nemoto, and L.C.L. Hollenberg. (2008). Deterministic optical quantum computer using photonic modules. *Physical Review A*, 78(3):32318.

[29] JM Taylor, H.A. Engel, W. Dür, A. Yacoby, CM Marcus, P. Zoller, and MD Lukin. (2005). Fault-tolerant architecture for quantum computation using electrically controlled semiconductor spins. *Nature Physics*, 1(3):177–183.

[30] B. T. H. Varcoe and P. J. Blythe. (2006). A cavity-QED scheme for cluster-state quantum computing using crossed atomic beams. *New J. Phys.*, 8:231.

[31] P. Walther, K.J. Resch, T. Rudolph, E. Schenck, H. Weinfurter, V. Vedral, M. Aspelmeyer, and A. Zeilinger. (2005). Experimental one-way quantum computing. *Nature*, 434(7030):169–176.